# Une approche formelle de description et de manipulation des objets structurés mathématiques

Bernard Fotsing Talla, Georges-Edouard Kouamou

# Une approche formelle de description et de manipulation des objets structurés mathématiques

Bernard FOTSING TALLA

Département d'informatique
Université de Yaoundé I
B.P 812 Yaoundé
CAMEROUN

bfotsing@yahoo.fr

Georges Edouard KOUAMOU

Département de Génie Informatique
École Nationale Supérieure Polytechnique
B.P 8390 Yaoundé
CAMEROUN

georges_edouard@yahoo.com

**RÉSUMÉ.** Nous présentons dans ce papier une approche formelle de description, d'affichage et de manipulation des objets structurés mathématiques ; basée sur le formalisme des grammaires attribuées. Nous nous intéressons particulièrement au problème d'affichage bidimensionnel et bidirectionnel de certaines expressions et formules mathématiques. En effet, en plus du caractère bidimensionnel que présentent certains symboles comme la racine carrée ou la matrice, on note le problème d'affichage de droite à gauche d'un texte arabe dans un contexte prévu pour un affichage de gauche à droite d'un texte indo-européen, ou encore un affichage bidirectionnel mélangeant les deux modes. Après une étude de quelques méthodes proposées dans la littérature, nous montrons comment la méthode des grammaires attribuées s'adapte facilement à ces types de problèmes.

**ABSTRACT.** We present in this paper a formal approach of description, posting and handling of the mathematical structured objects; based on the formalism of attribute grammars. We are interested particularly in the problem of two-dimensional and bidirectional posting of certain expressions and mathematical formulas. Indeed, in more of the two-dimensional character that presents certain mathematical symbols like the square root or the matrix, we also note the problem of posting right-to-left of an Arab text in a context planned for a posting left-to-right of an Indo-European text, or a bidirectional posting mixing the two modes. After a study of some solutions suggested in the literature, we show how the method of attribute grammars adapts easily to these types of problem..

**MOTS-CLÉS :** Grammaires attribuées, évaluation incrémentale des attributs, formules mathématiques, DTD, MathML, XML, interaction homme machine, CACEDE.

**KEYWORDS:** Attribute grammars, incremental evaluation of attributes, mathematics formulas, DTD, MathML, XML, man-machine interface, CACEDE.

**Novembre 2005, Numéro spécial CARI'04**

**Revue ARIMA** **pages 71 à 86**

## 1. Introduction

De plus en plus, les systèmes d'information documentaire évoluent et deviennent complexes et volumineux. A côté des bases de documents classiques et traditionnelles, notamment l'interconnexion des bibliothèques publiques, ou des centres de documentation d'entreprises coopérantes, le développement du Web apparaît comme l'exemple le plus connu de base de documents hypermédias répartis. La conception et la diffusion des documents scientifiques complexes restent encore aujourd'hui un défi majeur posé aux chercheurs compte tenu de la diversité socioculturelle des potentiels utilisateurs. Des concepts et standards de représentation et de manipulation de ces informations hétérogènes sont développés afin de faciliter leur échange entre applications, et leur intégration sur internet.

C'est dans cette optique qu'a été initié un projet de conception d'un atelier collaboratif d'édition des documents électroniques (CACEDE) dont le but à long terme se résume en deux points principaux :

- sur le plan de la recherche, il s'agit d'étudier les techniques de métaprogrammation (programmation par aspects, programmation intentionnelle, supercompilation) à l'aide des outils et méthodes empruntés à la compilation classique. Plus précisément, le formalisme des grammaires attribuées ou de généralisations raisonnables de celles-ci nous semble être un cadre bien adapté à la mise en place de ces différentes techniques dont l'intérêt principal est la réutilisation de codes et la spécialisation de programmes.

- sur le plan pratique, l'outil développé a pour objectif principal de fournir un atelier de développement de documents électroniques publiables sur le Web (par exemple des supports de cours pour un système d'EAD - Enseignement A Distance) conçu de manière collective par un groupe d'auteurs ; avec la possibilité à tout moment d'extraire des versions papier (polycopiés) utilisant le format LaTeX et donc d'une qualité typographique plus agréable. Il s'agit de concevoir et de mettre en oeuvre un éditeur structuré permettant à un groupe d'auteurs de créer de manière collective des documents électroniques pouvant intégrer de manière récursive du texte, des tableaux, des images, des schémas vectoriels, des formules mathématiques, etc..

## 2. Problématique

Le problème posé initialement implique plusieurs éléments : l'édition, la collaboration, l'évaluation incrémentale, la persistance. Il est donc d'une complexité trop importante pour que nous puissions le traiter entièrement dans cet article. Ainsi, nous nous sommes intéressés à un fragment significatif de la syntaxe (documents

structurés mathématiques) tout en essayant de tirer le meilleur parti des outils et résultats existants et disponibles dans le domaine.

La création de documents mathématiques pose plusieurs types de problèmes, notamment en ce qui concerne les objets mathématiques. Il faut résoudre entre autres le problème d'interface de saisie et de visualisation des équations et autres symboles mathématiques ; donner un sens à la représentation de ces objets ; trouver un format adéquat pour le stockage de tels documents ; etc. Le traitement des objets mathématiques soulève plusieurs autres problèmes ; on peut citer entre autres, la complexité et la diversité des règles typographiques de gestion des grandes formules [10], l'efficacité et l'incrémentalité des algorithmes de formatage, la sélection des sous-expressions, l'ambiguïté des notations mathématiques, etc. L'exemple le plus palpable de ces problèmes est le dessin des délimiteurs [, {, ∫, ∑, etc. dont la taille varie en fonction de l'expression mathématique. En plus de ce caractère bidimensionnel et incrémental de certains symboles mathématiques, il faut aussi résoudre le problème d'affichage bidirectionnel dans les documents scientifiques multilingues. Par exemple l'affichage du texte arabe (de droite à gauche) dans un contexte prévu pour un affichage de texte indo-européen (de gauche à droite) ou encore l'affichage bidirectionnel mélangeant les deux modes (texte arabe incluant du texte indo-européen ou des formules mathématiques)[9]. Nous nous intéressons particulièrement à ce dernier aspect dans cet article où nous proposons dans la section 4 une approche qui s'appuie sur le formalisme des grammaires attribuées.

## 3. Manipulation des objets structurés mathématiques.

L'universalité de la science combinée à la diversité culturelle de la population mondiale entraîne un besoin sans cesse croissant et pressant d'outils efficaces permettant de manipuler et d'afficher les documents scientifiques dans plusieurs langues incluant du texte et des formules mathématiques. Ceci implique une gestion automatique des changements de direction, puisque toutes les langues ne s'écrivent pas et ne se lisent pas de la même façon ou dans la même direction. En effet l'arabe et l'hébreu sont écrits de la droite vers la gauche dans le sens horizontal, le japonais et le coréen sont écrits de la droite vers la gauche dans le sens vertical descendant. Le français, l'anglais et toutes les autres langues africaines et indo-européennes sont écrits et lus de la gauche vers la droite dans le sens horizontal.

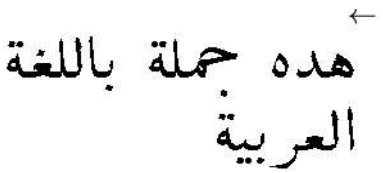


**Figure 1.** *Exemple de texte arabe*


**Figure 2.** *Exemple de texte japonais*

On remarque ainsi qu'en plus du problème classique de bi-dimension que posent certaines formules mathématiques (racine, matrice, puissance, …), l'aspect multilingue des documents scientifiques pose un autre problème ; celui de bi-direction. Ce problème devient crucial lorsqu'il faut manipuler à la fois des objets orientés de droite à gauche (par exemple suivant le modèle arabe) et des objets orientés de gauche à droite (par exemple suivant le modèle latin) dans le même document [8]. En effet, contrairement à un document mathématique (composé de formules et du texte) qui est orienté de droite à gauche dans le style égyptien, le texte du même document dans le style marocain ou algérien sera écrit de droite à gauche tandis que les formules seront écrites de gauche à droite. Des algorithmes et des outils existent dans la littérature et permettent d'afficher de façon bidirectionnelle des objets dans un même document.

Le système ArabTEX [7] qui fait partie des outils non interactifs de formatage multilingue résulte des travaux effectués par Knuth [6]. Ce dernier a développé des algorithmes pour le formatage du texte linéaire mélangeant du texte écrit de droite à gauche et du texte écrit de gauche à droite.

L'algorithme bidirectionnel d'Unicode [14] représente un standard général pour traiter le changement de direction dans un texte linéaire. Cependant, il prend uniquement en compte le changement de direction du texte ordinaire sans objets complexes tels que les formules mathématiques.

Dans sa thèse, Naciri [8] s'est particulièrement intéressée à l'étude du cas des documents scientifiques contenant des formules mathématiques écrites dans le sens inverse du texte arabe (style marocain et algérien). La technique utilisée est basée sur la description d'un langage abstrait, appelé PPML[1] [4] proposé par le système CENTAUR pour exprimer l'affichage des objets structurés. Le PPML permet de transformer les

[1] Pretty Printing Meta Language

objets représentés sous forme d'arbre de syntaxe abstraite en un arbre de boites correspondant à l'affichage à l'écran. Dans cette approche, le moteur d'affichage FIGUE est étendu pour prendre en compte l'affichage bidirectionnel. Le but est de fournir un composant d'affichage bidirectionnel permettant de manipuler et de mélanger des formules et du texte écrits dans les deux sens (droite-gauche et gauche-droite) en prenant en compte la nature bidimensionnelle des formules mathématiques. Pour y parvenir, le langage de boîtes d'affichage décrit pour les cas d'affichage unidirectionnel est modifié en introduisant de nouvelles notations pour définir la direction d'affichage de ces boîtes. Chaque boîte graphique a donc son propre algorithme de formatage adapté au contexte bidirectionnel, puisque les formules mathématiques (bidimensionnelles pour certaines) ne se comportent pas de la même façon que du texte simple (unidimensionnel). Chaque boîte a aussi un indicateur de direction ; et celui-ci peut prendre comme valeur Gauche-vers-Droite, ou Droite-vers-Gauche ou Neutre. Les algorithmes complets et détaillés de ces techniques d'affichage bidirectionnel des objets structurés dans les documents multilingues sont décrits au chapitre 6 de [8].

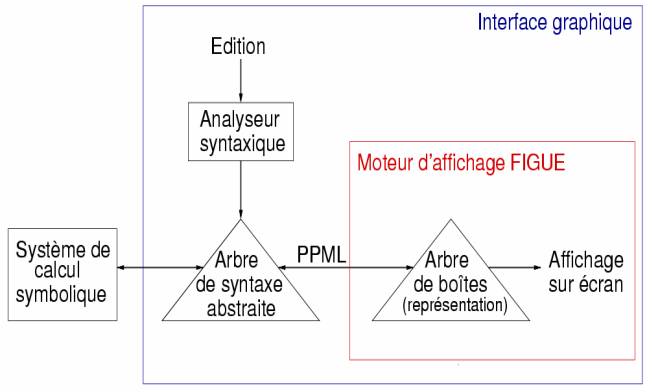


**Figure 3**. *Schéma général des interfaces graphiques utilisant FIGUE pour les systèmes de calcul symboliques (extrait de [8])*

# 4. Une approche basée sur le formalisme des grammaires attribuées

## 4.1. Rappel sur les grammaires attribuées

Plusieurs méthodes et techniques d'analyse et de spécification ont été étudiées pour la traduction des langages de programmation et l'implantation des applications d'édition de documents. Si la technologie orientée objet est la plus utilisée actuellement, la méthode des grammaires attribuées constitue l'un des formalismes les plus importants pour la spécification et l'implémentation des éditeurs structurés. Depuis leur introduction en 1968 par Knuth, elles ont été largement étudiées et continuent à être l'objet de nombreux travaux de recherche.

Une définition dirigée par la syntaxe est la donnée d'une grammaire algébrique dans laquelle chaque symbole grammatical est associé à un ensemble d'attributs partitionnés en *attributs hérités* et en *attributs synthétisés*. De plus, un ensemble de règles (dites *règles sémantiques*) est associé à chaque production et indiquent les dépendances entre les valeurs des attributs des symboles grammaticaux qui apparaissent dans la production.

Soit un nœud $u$ d'un arbre de dérivation associé à un symbole grammatical $X$ ($u$ est dit une occurrence de $X$). Ce nœud a les mêmes attributs que $X$. Intuitivement, la valeur d'un attribut synthétisé d'un nœud ne dépend que du sous-arbre issu de ce nœud, tandis qu'un attribut hérité de ce nœud ne dépend que du reste de l'arbre (le contexte). Ainsi donc, si $p : X_0 \rightarrow X_1 \ldots X_n$ est une production, on appelle attribut d'entrée de la production $p$ tout attribut qui est soit un attribut hérité de $X_0$ (dont la valeur doit provenir du contexte), soit un attribut synthétisé d'un des $X_i$ pour $1 \leq i \leq n$ (dont la valeur doit provenir du sous-arbre correspondant). Les autres attributs, c'est-à-dire les attributs synthétisés de $X_0$ et les attributs hérités des $X_i$ pour $1 \leq i \leq n$ sont appelés attributs de sortie ou attributs définis de la production. Les règles sémantiques associées à la production $p$ contiennent exactement une définition de la forme $a = f(b_1, \ldots, b_k)$ pour chaque attribut de sortie $a$ de $p$ dans laquelle les $b_i$ sont des attributs d'entrée de $p$. La fonction $f$ peut avoir des "effets de bord" (affectation d'une variable globale). Si $f$ se contente de produire des effets de bord (c'est à dire que $a$ est un attribut fictif dont la valeur n'est pas utilisée) on parlera d'action sémantique. Si, au contraire, les fonctions sémantiques n'ont aucun effet de bord, on dira que la définition dirigée par la syntaxe est une grammaire attribuée.

Le principe des grammaires attribuées peut donc se résumer ainsi qu'il suit : on considère une grammaire indépendante du contexte. A chacun de ses non-terminaux, on attache deux ensembles de symboles, les attributs synthétisés, qui véhiculent l'information depuis les feuilles d'un arbre de dérivation jusqu'à la racine, et les attributs hérités, qui transportent l'information en sens inverse. A chaque production on

associe un ensemble de règles sémantiques qui spécifient comment calculer les attributs de sortie ; c'est-à-dire les attributs synthétisés du non-terminal membre gauche de la production et les attributs hérités des non-terminaux du membre droit de la production, en fonction des attributs d'entrée de la production (les autres attributs).

## 4.2. Méthodologie générale adoptée

Une DTD[2] (spécialement les DTD normalisées) a souvent besoin d'être modifiée pour lui rajouter de nouvelles extensions, c'est le cas précisément de MathML qui est une grammaire XML pour représenter les formules mathématiques. Le but de notre travail n'a pas été de créer une DTD générique propre, ni de prendre en compte toute la DTD complète de MathML (dialecte XML pour les formules mathématiques), mais nous en avons extrait une partie (qui soit utilisable pour valider notre approche) et nous l'avons étendue en ajoutant quelques attributs (au besoin) à certains de ses éléments. Ainsi, nous partons d'une description par cette "nouvelle" DTD en MathML de la structure logique d'un document mathématique. Ce qui procure d'une part une syntaxe abstraite sous la forme d'un type algébrique, et associe d'autre part, à chaque type (catégorie syntaxique) un ensemble d'attributs. Un aspect pour cette structure logique de documents est une grammaire attribuée non ambiguë et non circulaire [3]. Un environnement d'édition est alors constitué d'une valuation incrémentale pour l'ensemble (éventuellement vide) des attributs synthétisés de l'axiome.

Un document structuré est donc représenté en mémoire par un arbre attribué conforme à une DTD qui est modifié suite à une interaction de l'utilisateur. La structure d'un document mathématique (incluant ses représentations logique et physique) se laisse donc naturellement représenter sous la forme d'un arbre décoré conforme à une DTD MathML. Les différents aspects de cet arbre décoré (forme graphique pour affichage à l'écran pendant l'édition, forme $LAT_EX$ pour impression sur support papier, forme HTML pour publication sur le WEB, systèmes d'annotations et gestion des versions pour l'édition collaborative) sont associés à des familles d'attributs inter-dépendants. XML (MathML pour notre cas) est utilisé comme langage associé à la représentation intentionnelle. Chaque interaction de l'utilisateur (pression sur une touche du clavier, clic de la souris, …) correspond à une modification de l'arbre de dérivation. Ce qui conduit à une révaluation partielle des attributs des nœuds (évaluation incrémentale).

## 4.3 Exemple d'un document mathématique multilingue

Considérons la phrase suivante formée d'un texte en français (gauche-droite), d'une expression mathématique, terminée par un texte en arabe (droite-gauche) ; le tout forme un document mathématique multilingue :

[2] Document Type Definition

*L'expression* $\frac{3}{x}$ صحيحة

La structure logique d'un tel document structuré est décrite par notre DTD MathML [15] (en utilisant bien sûr la syntaxe XML) ainsi qu'il suit :

```
<!ELEMENT mfrac (mrow , mrow) >
        <!ATTLIST mfrac %att-globalatts; 'numvalue CDATA #IMPLIED
                        denomvalue CDATA #IMPLIED
                        linethickness CDATA #IMPLIED
                        numalign (left | center | right) #IMPLIED
                        denomalign (left | center | right) #IMPLIED
                        rowdir (LR |RL | NULL) #IMPLIED
                        columndir (HL |LH | NULL) #IMPLIED '>
<!ELEMENT mtext (#PCDATA) >
        <!ATTLIST mtext %att-globalatts;
                        %att-fontinfo;
                        'value CDATA #IMPLIED
                        rowdir (LR |RL | NULL) #IMPLIED
                        columndir (HL |LH | NULL) #IMPLIED ' >
```

où *%att-globalatts* est une entité représentant l'ensemble des attributs partagés par tout élément d'un document structuré mathématique comme la largeur (*width*), la hauteur (*height*), la position de l'élément par rapport au bord supérieur, *top* (respectivement au bord gauche, *left*) de l'écran, etc. Le document XML correspondant et conforme à cette DTD va se présenter comme suit :

```
<?xml version="1.0" encoding= "ISO-8859-1" standalone="no"?>
<!DOCTYPE mathDocument SYSTEM "mathml.dtd">
<mathDocument>
<title>Un exemple de document multilingue</title>
<math>
        <mtext>L'expression </mtext>
        <mfrac>
        <mrow><mn>3</mn></mrow>
        <mrow><mi>x</mi></mrow>
        </mfrac>
        <mtext> صحيحة </ mtext>
</math>
</mathDocument>
```

Cette structure logique sous forme de DTD procure une syntaxe abstraite sous la forme d'un type algébrique auquel nous associons à chaque type un ensemble d'attributs. Nous en déduisons une grammaire attribuée non ambiguë et non circulaire dont les symboles non-terminaux forment un sous-ensemble des catégories syntaxiques de cette structure logique; et dont les attributs d'un non-terminal (de la grammaire attribuée) forment un sous-ensemble des attributs de la catégorie syntaxique qui lui est associée (dans la DTD). Nous donnons ci-dessous la grammaire attribuée associée à l'exemple du document mathématique multilingue décrit précédemment.

Pour le premier élément (texte en français *« L'expression* ») on aura :
*mtext → #PCDATA*

*mtext.value = #PCDATA*

*mtext.cursor = b*

*mtext.width = length (#PCDATA)*

*mtext.height = 1*

*mtext.rowdir = 'LR' /*sens de lecture du texte en français: horizontal gauche à droite*/*

*mtext.columndir = NULL /*pas de lecture en vertical */*

*mtext.display (mtext , value) /* aspect affichage interactif à l'écran */*

Pour le deuxième élément (la fraction 3/x), on aura :
*mfrac → mrow mrow*

*mfrac.numvalue = mrow1.value /* numérateur */*

*mfrac.denomvalue = mrow2.value /* dénominateur */*

*mfrac.cursor = b /* présence ou non du curseur d'édition sur ce noeud*/*

*mrow1.top = mfrac.top /* position par rapport au bord supérieur du numérateur */*

*mrow1.left = mfrac.left /* position par rapport au bord gauche du dénominateur */*

*mrow2.top = mfrac.top +mrow1.height*

*mrow2.left = mfrac.left*

*mfrac.linethickness = 1 /* épaisseur de ligne de la barre de fraction */*

*mfrac.numalign = 'left' /* alignement du numérateur */*

*mfrac.denomalign = 'left' /* alignement du dénominateur */*

*mfrac.width = max (mrow1.width, mrow2.width) /* largeur de la fraction */*

*mfrac.height=mrow1.height+mrow2.height+mfrac.linethickness /* sa hauteur*/*

*mfrac.rowdir = 'LR' /*sens de lecture de la fraction : horizontal de gauche à droite*/*

*mfrac.columndir = NULL /*pas de lecture en vertical */*

*mfrac.display(mfrac.value, mfrac.linethickness,mfrac.numalign, mfrac.denomalign)*

Pour le dernier élément (texte en arabe), on aura :
*mtext → #PCDATA*

*mtext.value = #PCDATA*

*mtext.cursor = b*

*mtext.width = length (#PCDATA)*

*mtext.height = 1*

*mtext.rowdir = 'RL' /*sens de lecture du texte arabe : horizontal de droite à gauche */*

*mtext.columndir = NULL /*pas de lecture en vertical */*

*mtext.display (mtext.value) /* aspect affichage interactif à l'écran */*

Créer un tel document en utilisant un éditeur structuré implique une croissance graduelle d'un arbre de dérivation c'est-à-dire le développement progressif des symboles terminaux de la grammaire attribuée. L'arbre de dérivation qui en résulte est entièrement décoré (attribué). La figure ci-dessous est la représentation interne du document mathématique exemple en mémoire.

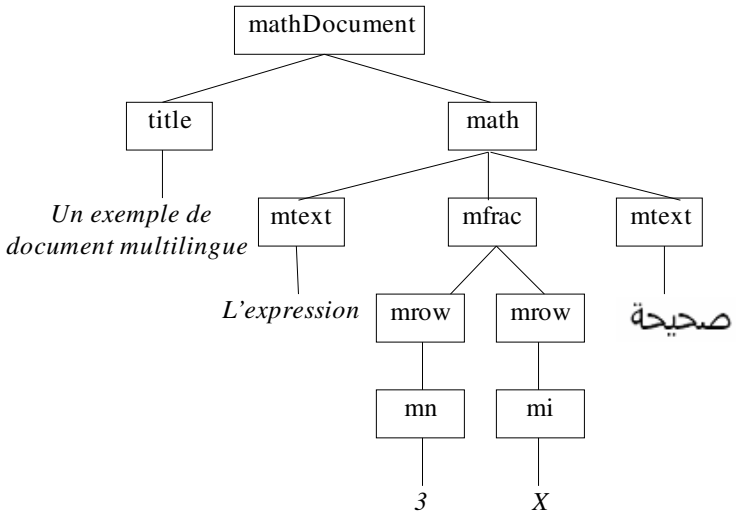


**Figure 4.** *Arbre de dérivation du document exemple*

Par des calculs simples et appropriés de certains attributs de cet arbre, on peut obtenir plusieurs aspects du document (attribut *display* pour l'affichage à l'écran, attribut *html* pour la version HTML du même document, etc.)

## 4.4. Principe de l'édition structurée

Un éditeur structuré est un outil qui construit un document en développant progressivement les symboles non-terminaux d'une grammaire attribuée [1]. Ainsi, créer un fichier (un document) en utilisant un tel éditeur implique une croissance graduelle de l'arbre de dérivation. Pendant le développement, un fichier est un arbre de dérivation partiel, c'est-à-dire qui contient des non-terminaux au niveau des feuilles (symboles non encore développés). Ceci est un problème potentiel car à un non-terminal non développé $X$, il n'y a aucun moyen de donner des valeurs aux attributs synthétisés de $X$, ou à n'importe lequel de ses successeurs. Pour éviter ce problème, on enrichit la grammaire en introduisant la notion de production complémentaire $X \rightarrow \perp$, pour chaque symbole non-terminal $X$ [13]. Le symbole $\perp$ signifie non développé, et les équations sémantiques de la production complémentaire $X \rightarrow \perp$, définissent les valeurs des attributs synthétisés de $X$. La modification d'un document implique la reconstruction d'un arbre de dérivation par application successive des opérations d'élagage et de greffe de sous-arbres. Cette suite d'opérations est connue sous le nom de *remplacement de*

*sous-arbres* [13]. Ainsi, le remplacement du sous-arbre *u* (dont la racine est *r*) par le sous-arbre *u'* (de racine *r'*) consiste à élaguer *u* et à greffer *u'* à sa place comme le montre la figure ci-dessous.

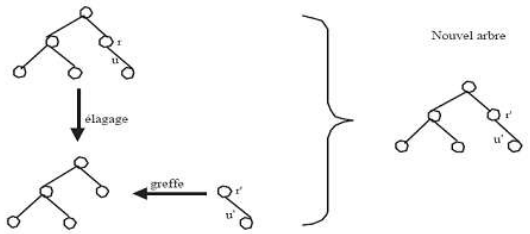


**Figure 5.** *Restructuration d'un arbre de dérivation par élagage et greffe*

A chaque stade d'édition, le curseur d'édition est positionné à un nœud interne de l'arbre de dérivation. Une session d'édition est alors vue comme une succession d'opérations de remplacements de sous-arbres et de mouvements du curseur d'édition qui commence à la racine de l'arbre sémantique complet et entièrement attribué. Cependant au point de modification, les valeurs des attributs peuvent ne plus être consistantes vis-à-vis des règles sémantiques de la grammaire attribuée qui les définissent. Ce qui rend le sous-arbre inconsistant et par conséquent l'arbre de dérivation tout entier. L'évaluation incrémentale a donc pour but de rétablir la consistance du nouvel arbre en ne recalculant que les instances d'attributs nécessaires [11] et [12] ; c'est-à-dire en ne révaluant que les instances d'attributs touchées par le remplacement de sous-arbres.

Par ailleurs, l'utilisateur qui construit son document en faisant grandir l'arbre de dérivation qui le représente, voit le document uniquement comme la représentation textuelle qu'il reçoit à l'écran. Il ne se soucie pas de l'arbre qui est manipulé en mémoire.

## 4.5. Description des algorithmes de calcul d'attributs

L'évaluation d'attributs est un procédé par lequel on calcule à l'aide des règles sémantiques de la grammaire attribuée toutes les instances d'attributs d'un arbre de dérivation [1]. Le caractère non procédural des grammaires attribuées est un avantage particulier qui a été à l'origine de nombreux travaux de recherche sur le calcul d'attributs. Comme il n'est pas spécifié dans quel ordre on calcule les attributs (mais seulement ce que l'on calcule), plusieurs façons d'évaluer les attributs ont été étudiées (évaluation séquentielle, évaluation parallèle, évaluation incrémentale).

Nous nous sommes inspirés de l'algorithme de calcul incrémental d'attributs proposé par Jourdan [5] qui transforme chaque attribut en une fonction. Cet algorithme simule le parcours du graphe qui décrit les dépendances entre les différentes instances d'attributs

dans l'arbre de dérivation décoré, et utilise la pile des appels entre les différentes fonctions. L'évaluation est alors faite à travers des appels récursifs entre les fonctions. Elle s'arrête lorsque toutes les instances d'attributs synthétisés de la racine de l'arbre de dérivation ont été évaluées. Cette façon d'effectuer les calculs a l'avantage d'être simple, de ne pas restreindre le type de grammaires attribuées à traiter et surtout, d'effectuer une évaluation par nécessité.

### 4.5.1 Calcul des attributs synthétisés

Pour un attribut synthétisé, la fonction possède un nœud de l'arbre de dérivation comme "unique" argument. Considérons l'exemple de la règle suivante (une production de la grammaire attribuée obtenue dans [3]) : *mfrac → mrow mrow*, alors la valeur de son numérateur sera calculée récursivement par la fonction suivante (nous utilisons la notation orientée objet proche de Java) :

*public Object getMfracNumValue(mfrac){ return getMnValue(mrow1);}*

et celle de son dénominateur par une fonction similaire :

*public Object getMfracDenomValue(mfrac){ return getMrow2Value(mrow2);}*

### 4.5.2 Calcul des attributs hérités

La fonction qui calcule la valeur d'un attribut hérité a pour arguments, le nœud considéré dans l'arbre, le nœud père de celui-ci et plusieurs autres arguments. En considérant la même règle *mfrac → mrow mrow*, l'origine supérieure du numérateur de la fraction sera donnée par la fonction suivante:

*public int getMrow1Top(mrow1,mfrac){ return getMfracTop(mfrac);}*

et l'origine supérieure du dénominateur par une fonction similaire :

*public int getMrow2Top(mrow2,mfrac,mrow1){return getMfracTop(mfrac) + getMrow1Height(mrow1);}*

## 5. Discussion

En comparaison à l'approche proposée par Naciri [8], on constate ici que l'arbre de boîtes n'existe plus ; les règles de transformation PPML sont "remplacées" par les règles sémantiques de chaque nœud. Celles-ci (règles sémantiques) décrivent les règles de calcul des différents attributs de l'arbre de syntaxe abstraite qui est désormais entièrement décoré. Ces attributs sont calculés au fur et à mesure que le document grandit par des algorithmes appropriés de calcul d'attributs et le résultat peut aussi être visualisé en même temps sur une sortie standard comme l'écran de l'utilisateur (aspect interactif et incrémental de l'approche). Cette technique se résume dans la Figure 6 où l'analyse syntaxique du document source se fait suivant la grammaire attribuée spécifiée

par sa DTD. L'arbre de syntaxe abstraite qui en résulte est la représentation interne même du document en mémoire. Chaque événement déclenché par l'action de l'utilisateur sur le document se traduit par une modification par l'application de l'arbre de dérivation en respectant les règles sémantiques de la grammaire attribuée qui le décrit. Cette modification se fait en respectant à la fois les deux invariants que sont : le calcul d'attributs (règles sémantiques de la grammaire attribuée) et la DTD de la représentation intentionnelle MathML. Un calcul adéquat d'attributs se fait en même temps et permet de rafraîchir la représentation textuelle reçue à l'écran. C'est un aspect de la représentation interne de l'arbre en cours d'édition.

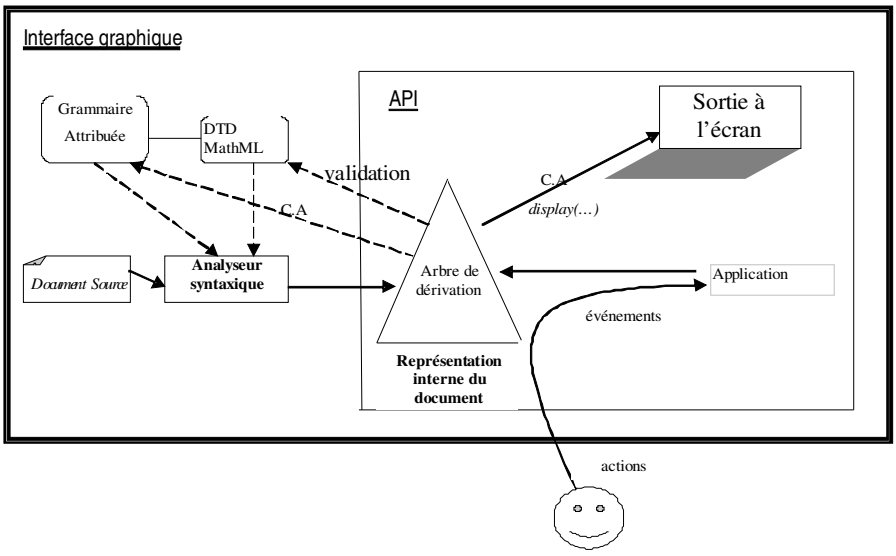


**Figure 6.** *Architecture fonctionnelle de l'édition structurée*

Le problème d'affichage bidirectionnel posé trouve une solution dans cet arbre de dérivation entièrement attribué. En effet, chaque nœud de l'arbre a un attribut *rowdir* pour indiquer le sens de lecture/écriture de chaque élément, c'est-à-dire si on lit ou écrit en horizontal de la gauche vers la droite (*LR* pour *left-right*) ou de la droite vers la gauche (*RL* pour *right-left*); et un autre attribut *columndir* pour indiquer si on lit ou écrit en vertical du bas vers le haut (*LH* pour *low-high*) ou du haut vers le bas (*HL* pour *high-low*). Des règles sémantiques appropriées sont définies pour le calcul de chacun de ces attributs à chaque nœud de l'arbre de dérivation. En combinant les valeurs de ces deux attributs dans le même arbre, on peut avoir dans un même document des textes et formules dans toutes les langues possibles écrits correctement (arabe, français, japonais,

et même des combinaisons de lecture/écriture inattendues). Ceci nous conduit (au-delà du bidirectionnel prévu) à un affichage multidirectionnel des objets structurés dans un même document.

## 6. Conclusion et perspectives

Nous avons traité de la manipulation formelle des objets structurés principalement en utilisant les grammaires attribuées. L'intérêt de la technique des grammaires attribuées réside dans le fait qu'on peut, à partir de l'arbre de dérivation complètement attribué, obtenir plusieurs aspects d'un même document. En effet, plutôt que d'afficher directement le résultat à l'écran, on peut considérer d'autres aspects du document comme une version $LaTeX$, une version Web et d'autres versions aux formats structurés. Il suffira pour les deux premiers cas de définir un attribut synthétisé *latex* pour la version $LaTeX$, un attribut synthétisé *mathml* (dans le cas des documents mathématiques) pour la version Web ; et par des calculs simples et appropriés d'attributs, on obtiendra les versions correspondantes du même document.

Les grammaires attribuées ont un inconvénient majeur d'être très volumineuses. A long terme nous envisageons de proposer un format permettant de les représenter de manière plus compacte.

Par ailleurs, FNC2 [2] est à l'heure actuelle un des évaluateurs d'attributs les plus étudiés pour les grammaires attribuées fortement non circulaires. Nous nous proposons d'expérimenter cet évaluateur, ceci en vue d'optimiser l'évaluation des attributs.

## 7. Bibliographie et biographie

### 7.1 Bibliographie

[1] CRUZ LARA SILVA S., 1988. GEODE: Un système pour la génération d'environnements de programmation intégrés, Thèse de Doctorat en Informatique, Institut National Polytechnique de Lorraine, novembre 1988.

[2] FNC2. http://www-rocq.inria.fr/oscar/www/fnc2/fnc2-fra.html

[3] FOTSING TALLA B., 2003. CACEDE : un éditeur structuré incrémental pour les formules mathématiques, Mémoire de DEA-Informatique, Université de Yaoundé I, Février 2003

[4] JACOBS I. and L; RIDEAU-GALLOT, 1994. The PPML Manual. Technical report, INRIA, 1994.

[5] JOURDAN M., 1984. An optional-time recursive evaluator for attribute grammars, INRIA Rocquencourt. Décembre 1984.

[6] KNUTH D. and P. MACKAY, 1987. Mixing right-to-left texts with left-to-right texts. TUG-boat, 8(1):14-25, April 1987.

[7] LAGALLY K., 1992. ArabTex –typesetting Arabic with vowels and ligatures. In J. Zlatuska, editor, EuroTex 92: Proceedings of the 7th European TeX Conference, pages 153-172, brno, Czechoslovakia, Sept. 1992. Masarykova Universita.
[8] NACIRI H., 2002. Conception et réalisation d'outils pour l'interaction homme machine dans les environnements de démonstrations mathématiques. Thèse de Doctorat en Sciences (Informatique) ; Université de Nice – Sophia Antipolis, décembre 2002.
[9] NACIRI H., REDEAU L., 2001. Affichage et manipulation interactive de formules mathématiques dans les documents structurés, Rapport de recherche n° 4140 – Mars 2001, Unité de recherche INRIA Sophia Antipolis.
[10] NACIRI H., REDEAU L., 2002. Affichage et diffusion sur Internet d'explications en langue arabe de preuves, Proceedings of the 6th CARI'02. Octobre 2002, Yaoundé.
[11] PARIGOT D., Mise en œuvre des grammaires attribuées : transformation, évaluation incrémentale, optimisations. Thèse de 3ème cycle, Université de Paris-Sud, Orsay, septembre 1987.
[12] PARIGOT D., Transformations, évaluation incrémentale et optimisation des grammaires attribuées : Le système FNC-2. PhD thesis, Université de Paris-Sud, Orsay, 1988
[13] REPS T., Generating Language-Based Environments, ACM Doctoral Dissertation Award 1993. The MIT Press, 1993
[14] The Bidirectional Algorithm, http://www.unicode.org/unicode/reports/tr9/bidi.
[15] VATTON I., 2000. W3C's Amaya 4.0 Editor Browser , 2000, W3C.

## 7.2 Biographie

M. FOTSING TALLA Bernard participe au projet CACEDE dans le cadre de sa thèse en Informatique (en co-tutelle entre l'Université de Yaoundé I et l'Université de Rennes I). Il s'intéresse aux techniques de compilation pour la manipulation des objets structurés.

M. KOUAMOU Georges Édouard est enseignant à l'Ecole Nationale Supérieure Polytechnique de Yaoundé. Il s'intéresse aux techniques de construction des environnements logiciels, et aux spécifications formelles basées sur les grammaires attribuées.